Real-World Post-Quantum Digital Signatures

Denis Butin[1], Stefan-Lukas Gazdag[2], and Johannes Buchmann[1]

[1] TU Darmstadt, Germany
{dbutin,buchmann}@cdc.informatik.tu-darmstadt.de
[2] genua mbH, Germany
stefan-lukas_gazdag@genua.eu

Abstract. Digital signatures are ubiquitous in modern security infrastructures. Their lack of diversity in industrial settings makes most contemporary systems susceptible to quantum computer-aided attacks. Alternatives exist, among which a family of well-understood schemes with minimal security requirements: hash-based signatures. In addition to being quantum-safe, hash-based signatures are modular, providing long-term security. They are not yet being used in practice. We discuss the reasons for this gap between theory and practice and outline a strategy to bridge it. We then detail our work to realise the described plan.

Keywords: Authenticity; Post-Quantum; Usability; Integration

1 Context and Motivation

Digital signatures are massively used in contemporary security infrastructures. A typical use case is the authentication of software updates, protecting the users of the software from the execution of malicious code. Another common application of digital signatures is server authentication in the Transport Layer Security (TLS) protocol, combined with the Hypertext Transfer Protocol in HTTPS.

Given the sheer amount and diversity of sensitive information transmitted online, robust security of digital signatures is essential. Security protocols such as TLS are typically not tied to any particular type of digital signature. For instance, the TLS protocol merely specifies functional requirements for its cipher suite. However, in practice, common security protocols are only instantiated with a small number of different signature schemes: RSA [43], DSA [19] and its elliptic curve variant ECDSA [31]. Due to the constant progress of cryptanalysis, this lack of diversity alone is already dangerous. Key size itself — assuming only brute-force attacks — is not the most worrying issue, since future computational power can be predicted with reasonable accuracy [33]. Nevertheless, more efficient and elaborate attacks may always emerge. Another more general issue will become increasingly problematic in the future: the classes of security assumptions of these three digital signature schemes.

Indeed, RSA, DSA and ECDSA all rely on the hardness of two mathematical problems: prime number factorisation and discrete logarithm computation. These two problems are susceptible to attacks using quantum computers. In fact, a 1994 algorithm by Shor [45] shows that the two aforementioned problems can be solved in polynomial time with respect to the input size on a quantum computer. Simply increasing key sizes will therefore not be a solution, and this exponential speed-up vis-à-vis classical computers will break RSA, DSA and ECDSA in practice.

While quantum computers capable of yielding such attacks do not yet exist, rapid progress is being made [14]. The energy storage time of quantum switches is increasing vastly [41]. Quantum bit storage time is also growing [44]. In addition, reports mention undisclosed work on quantum computers by governmental intelligence agencies [42]. Even under the optimistic assumption that progress will slow down and that the advent of quantum computers is still far-off, a precautionary approach to risk management demands post-quantum security infrastructures. In particular, post-quantum digital signatures schemes ought to be deployed on a large scale.

Fortunately, several types of post-quantum digital signature schemes already exist. Somewhat surprisingly, some of them have existed for decades but none of them enjoys widespread use. Often, initial performance and storage drawbacks have dampened practical use and interest. This paper focuses on a specific family of post-quantum digital signatures: hash-based signatures (HBS). Reasons for this choice, such as their modularity, are detailed in Sec.2. Several alternatives exist. All of them face obstacles to widespread deployment. Code-based signatures [13] suffer from excessively large space requirements for keys. More investigations into the operation and security of multivariate-based signatures [15] are needed. Lattice-based signatures [18] such as NTRUSign have been implemented and even patented. A number of strong attacks have been found [37], but since lattice-based signatures are continuously improving, they embody the best current post-quantum alternative to HBS.

Nevertheless, lattice-based signatures do not enjoy the advantages that HBS do. These are detailled in Sec. 2.

1.1 Contributions

The goal of this paper is to identify why post-quantum signatures are not being widely used yet, to outline a remedial strategy and to discuss current project work [21] to realise this plan. To this end, we start by describing HBS and arguments in favour of their use (Sec. 2). We then emphasise the numerous gaps still existing between theory and practice (Sec. 3), notably the shortage of concepts for dealing with stateful key management, the lack of standardisation and the absence of integration in cryptographic software libraries. The main part of the paper outlines a strategy to ready HBS for the real world and discusses our first steps to enact it (Sec. 4). Software integration, both in terms of core implementations in cryptographic libraries such as OpenSSL and in terms of corresponding protocol implementations, is an essential aspect. Standardisation efforts are described with an emphasis on strategic decisions. Parameter selection, use cases and performance improvement opportunities are discussed as well. We conclude with an outlook on future steps (Sec. 5).

2 Hash-based Signatures

We now recall the main properties of modern HBS. Their elementary building block is the use of a One-Time Signature (OTS) scheme. Many OTS key pairs are combined in a so-called *Merkle Tree* using a hash tree. Several improvements upon the plain Merkle construction have made it much more efficient. Modern HBS variants can be seen as the result of these three building phases, detailed below. We also emphasise important advantages of HBS with respect to other post-quantum signatures: their minimal security requirements and modularity.

2.1 OTS schemes

OTS schemes are digital signature constructions in themselves, but any OTS signature reveals part of the associated secret key. As a consequence, every OTS secret key can only be used once. Indeed, a second signature using the same key would reveal even more about the key, allowing further signatures to be forged. Two well-known OTS schemes are the Lamport-Diffie scheme [32] and the Winternitz scheme [16]. Their structure is similar, and their security requirement reduces to the collision resistance of the used hash function. In both cases, signing keys are randomly generated and the corresponding verification key is obtained by applying a one-way function (repeatedly, in the Winternitz case) to the signing key. As is usual for signature schemes, in both cases, signatures on a message are actually applied on the hash of the message.

The Winternitz OTS can be seen as a generalization of the Lamport-Diffie OTS. It features the so-called *Winternitz parameter*, which regulates a speed/size trade-off — between signature and key sizes on one hand, and between signature and verification time on the other hand. Lamport-Diffie offers no such flexibility. Modern HBS schemes all use either the Winternitz OTS, or one of its variants. The W-OTS$^+$ variant provides a tight security proof despite more modest security requirements on the underlying hash function. A resulting benefit is that W-OTS$^+$ is not threatened by a class of generic attacks (*birthday attacks*) that targets stricter security requirements [26]. For this reason, one can afford to securely use much shorter signatures with W-OTS$^+$.

2.2 Merkle trees

Since an OTS key pair can only be used once securely, several OTS key pairs must be combined in a single structure to yield a usable HBS scheme. This is realised in all HBS schemes by combining several OTS key pairs in one or more (binary) Merkle trees. A Merkle tree [35] is a complete tree relating several OTS key pairs to a single public key — the root of the Merkle tree. Merkle trees are also called hash trees because nodes are computed through hashing and concatenation. The leaves of the tree are the hashed OTS verification keys (i.e. OTS public keys). Signatures contain both an OTS signature and a way to authenticate this OTS signature against the public key of the entire tree.

2.3 Subsequent improvements

Several improvements upon the plain Merkle scheme have taken place over the last decade [10–12, 29]. The two most advanced of them are XMSS (the eXtended Merkle Signature Scheme) [10] and XMSS^{MT} (Multi Tree XMSS) [29]. We do not go into technical detail here but explain the main conceptual developments.

A major disadvantage of the initial proposal was the very large signing key size. Successive improvements have curbed this issue. In particular, the use of multiple hash trees (arranged in layers) to combine more OTS key pairs into a single structure enables an amount of OTS key pairs so large as to be plentiful for all practical purposes (e.g. 2^{80} OTS key pairs). OTS signing key storage requirements are dramatically reduced by using a pseudo-random number generator to obtain them in succession. It is then sufficient to store the generation seed value instead of all OTS signing keys. In some variants, signature generation times are reduced by distributing OTS generation and other computations evenly across steps. In schemes such as XMSS [10], it is possible to reduce the requirement on the underlying hash function. Shorter hash length requirements can then be used. Fig. 1 depicts an XMSS Merkle tree.

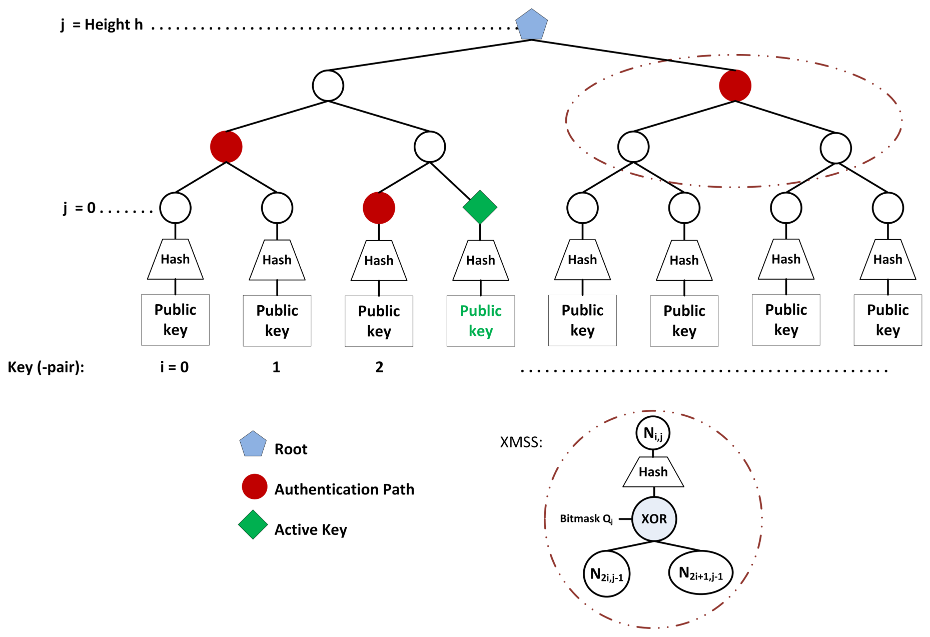

Fig. 1. An XMSS Merkle tree, following Hülsing's [25] description.

2.4 Forward security

HBS schemes can provide a valuable property: forward security [4]. The fact that schemes like XMSS or XMSS^{MT} can be forward secure means that all signatures generated before a key compromise remain valid. This attribute supports a form of long-term security.[3] Note that forward security is not an intrinsic property of HBS; it requires special constructs such as the use of a pseudo-random number generator. Since these constructs also provide other advantages (such as reduced storage requirements), there is much incentive to use them.

2.5 Further advantages of hash-based signatures

HBS enjoy a number of other advantages that motivate their deployment. A key aspect is the minimality of their security requirements. Indeed, all types of digital signatures require a secure hash function; yet, in the case of HBS, this requirement is sufficient to ensure overall security. This lack of additional assumptions reduces the security trust requirements.

Furthermore, the OTS scheme that forms the elementary building block of a HBS scheme — be it Lamport-Diffie or Winternitz — is not tied to any specific hash function. The chosen hash function only needs to be secure. This advantage is significant, because it means that HBS schemes are like templates: they may be instantiated with any secure hash function without changing their overall structure. Since hash functions have limited lifespans, this contributes to the longevity of HBS. It is an argument for long-term security: implementations of HBS schemes can be modified to use a more modern hash function without major structural modifications.

The reliance of HBS schemes on a secure hash function works as an early warning system [9]: once attacks on a hash function begin to emerge, a significant amount of time passes until the hash function's security is completely broken. This is especially true when an advanced OTS scheme such as W-OTS^{+} is used. The kind of attack that threatens W-OTS^{+} only emerges after attacks on stricter security requirements appear.[4] When first attacks occur, HBS schemes can be updated using a better hash function without major infrastructure modifications. By contrast, for other (non-modular) signature schemes, an attack may entirely compromise the scheme, requiring a switch to a completely different scheme. Such a change is more cumbersome than the mere replacement of a hash function.

[3] While the full descriptions of XMSS and its multi-tree variant are provably forward secure [10, 29], this property relies on constructs irrelevant to interoperability. The proposed standardisation of HBS schemes we discuss later therefore does not intrinsically yield forward security, but permits it if the right components (such as a forward secure pseudo-random number generator) are used in its implementation.

[4] When using W-OTS^{+}, security requirements are reduced from collision resistance to second preimage resistance.

3 Obstacles to Widespread Use

It can be seen from the previous section that the theory underlying HBS is well-understood. Moreover, subsequent improvements have solved the performance issues that were an obstacle for more primitive HBS schemes. Even proof-of-concept implementations exist. However, we contend that many steps are still needed if HBS are to become widely used. This section describes the main roadblocks. In the remainder of the paper, we will present a detailed strategy to overcome the outstanding issues.

3.1 Statefulness

Since HBS always depend on OTS schemes, they must keep track of which OTS signing keys have already been used. Using an OTS signing key more than once is insecure. A global key index is therefore used. When a signature is generated, an updated secret key is output too.

As a consequence, HBS schemes are said to be *stateful*. Statefulness is inconvenient in practice for a number of reason. First, it does not fit common software interfaces. Second, performance is impacted by frequent key access. Besides, key storage conditions become critical. Copies or backups containing keys, and old key states in general, must be avoided for security reasons. These concrete consequences need to be taken into account when developing mitigation strategies. Dealing with statefulness is an integral part of the solutions described later (Sec. 4), even though we focus on more high-level issues in this paper. Stateless HBS, based on random index selection and a few-time signature scheme, have recently been introduced [5] to circumvent this issue. We stick to stateful HBS for performance reasons. Another argument in their favour is forward security.

3.2 Lack of standardisation

All currently popular digital signature schemes have been standardised. RSA, DSA and ECDSA are standardised in the Digital Signature Standard by NIST [36]. Stakeholders prefer to use standardised security techniques. The standardisation process guarantees increased investigation by experts. For instance, the scrutiny resulting from the selection process for the SHA-3 hash function [6] has resulted in over a dozen of cryptanalytic papers on Keccak [8, 17]. Standards usually include security parameter recommendations. This point is especially important for cryptographic schemes that use numerous different parameters. While an abundance of parameters normally allows for greater flexibility and trade-off opportunities, it also considerably increases the complexity of the decision process and makes unforeseeable consequences more likely. Typical parameter sets, especially when coupled with descriptions of typical use cases, offer stakeholders practical, ready-to-go solutions with established guarantees. Additionally, standards are more ergonomic than research publications for means of implementation. The usual tone of standards requires a level of explicitness that is not always found in research literature, but which is indispensable for implementers.

Research-oriented descriptions of cryptographic schemes may also use abstract constructions without clarifying which concrete methods may be used to instantiate them. For instance, the initial description of XMSS [10] prescribes the use of a pseudo-random function family. In practice (i.e. for concrete implementations), a secure hash-function provides the functionality of this function family.

3.3 Missing availability in commonly used software libraries

Stakeholders should not be expected to adopt new schemes, whatever their advantages, if system integration is too experimental. In the case of HBS, implementations exist [10, 27], but they are standalone. While such proof-of-concept implementations are an important, necessary step towards widespread use, it is inopportune for organisations to create their own, specific ad hoc implementations. Avoiding case-by-case implementation of cryptographic primitives is common advice. Instead, well-known and tested cryptographic libraries ought to be used. These libraries include abstractions to facilitate system integration and combination. A well-known example is OpenSSL [1], an open-source library providing not only core implementations of cryptographic primitives, but also their combination in commonly used protocols such as TLS.

4 Bridging the Gap

Standardisation, integration into common cryptographic libraries and the definition of typical use cases and parameters are expected to support the adoption of HBS by increasing clarity, providing interoperability and decreasing costs for organisations.

4.1 Standardisation

An Internet-Draft for the basic Merkle HBS scheme exists [34]. To make further improvements to the scheme widely available, an extension covering the techniques named above is desirable. Therefore, a standard covering the main aspects of newer HBS variants is necessary. We are contributing to an IETF Internet-Draft [28] covering XMSS and its multi-tree variant XMSS^{MT}. Some strategic considerations are explained in the current section.

As the underlying OTS scheme, we picked W-OTS^{+} [26]. As explained earlier, all current efficient OTS schemes are variants of Winternitz's initial proposal. The reason for this choice are the shorter signatures obtained when using W-OTS^{+}, together with improved security guarantees.

The standard proposal will include the pivotal algorithms for XMSS and XMSS^{MT}. Regarding XMSS, functions for signing, verification and public key generation are defined. In addition, using the previous one, a function is defined to construct tree leaves from OTS public keys. Another crucial auxiliary algorithm computes the inner nodes. These two functions, in turn, are building blocks of the algorithms for signature generation, checking and public key computation. The use of a pseudo-random number generator for key generation was mentioned in Sec. 2. This aspect is not part of the standard, since its scope is limited to features relevant to interoperability.

XMSS is a building block of XMSS^{MT}. As a consequence, the algorithms for XMSS^{MT} signing, verification and public key generation in the standard proposal use the corresponding XMSS algorithms as subroutines. Normally, XMSS^{MT} supports the selection of a different Winternitz parameter on every layer. The Winternitz parameter determines how many message bits are signed simultaneously. In the standard proposal to which we contribute, a single Winternitz parameter is used for the entire XMSS^{MT} scheme. The reason for this choice is the resulting simplification of implementation. The cost in terms of flexibility is modest. Similarly, XMSS^{MT} normally allows a different tree height for each layer, but we proscribe a global tree height used for all layers, for the same reasons. Since the SPHINCS stateless HBS uses XMSS as a constituent, the planned standard will also act as a first step towards the standardisation of such schemes. The standardisation of both categories of HBS schemes is beneficial since different use cases will take advantage of their respective strengths.

4.2 Integration in cryptographic libraries

To make modern HBS schemes widely available, our strategy is to integrate them with three important security protocols:

- TLS, to be used with transport layer protocols — e.g. with HTTP for HTTPS;
- SSH, e.g. for remote login;
- S/MIME, for email authentication.

All of these security protocols are supported by common cryptographic software libraries. For TLS and SSH integration, we plan to use OpenSSL and OpenSSH. OpenSSL is widely known and used. While serious vulnerabilities have recently been discovered in OpenSSL [39,40], it is being revisited more closely as a result. In addition, forks such as LibreSSL [38] and BoringSSL [22] have emerged; synergy effects between these projects are expected to further tighten implementation security. Since OpenSSH uses OpenSSL, some of the necessary work benefits both the TLS and the SSH integration.

OpenSSL consists of two sub-libraries: a collection of core implementations of cryptographic constructs, and a library implementing the TLS protocol, relying on the other one. Integrating HBS with OpenSSL is therefore a two-step process. We start by providing a core implementation. Once this is done, further work is required to adapt the protocol-level module so the TLS protocol can actually be instantiated with a cipher suite using HBS for authentication. As far as the core implementation is concerned, existing HBS implementations serve as a starting point but cannot be used as is. OpenSSL defines a number of abstractions, such as the EVP high-level interface to cryptographic functions and other APIs. These abstractions must be extended to cope with the peculiarities of HBS. Notably, changes to key handling are required due to the statefulness of HBS.

When integrating HBS in a TLS cipher suite, a natural question is whether an entire cipher suite can be built. Generally speaking, primitives using symmetric cryptography are not significantly threatened by quantum computers because Shor's algorithm [45] cannot be used against them. Symmetric constructions are somewhat vulnerable to another quantum algorithm, by Grover [23]; but the resulting speed-up is much less threatening. Doubling the key length is sufficient to counteract the threat and to keep the same security level as before. As a consequence, the symmetric primitives used in TLS cipher suites do not need to be replaced for the cipher suite to be post-quantum as long as key length choices take Grover's algorithm into account. However, the key exchange part ought to be replaced. It typically relies on the Diffie–Hellman method, which relies on the discrete logarithm problem and is therefore susceptible to attacks based on Shor's algorithm. Fortunately, post-quantum key exchanges have been put forward. In particular, a key exchange based on the Ring Learning With Errors problem has already been demonstrated in an OpenSSL fork implementation [7]. Combining this implementation with an HBS-based key exchange will yield a post-quantum TLS cipher suite.

The S/MIME standard defines encryption and signing of email messages using the MIME format. S/MIME is also available in OpenSSL, but we plan to integrate HBS with this protocol using a different cryptographic software library: Bouncy Castle [2]. As in OpenSSL, a prerequisite to security protocol integration (S/MIME in this case) is the availability of a HBS as a core implementation. Some preliminary work already exists here, since the GMSS HBS scheme was implemented in Flexiprovider [46], a set of cryptographic modules for the Java Cryptography Architecture. Flexiprovider was later integrated into Bouncy Castle. A more advanced scheme than GMSS is suitable for S/MIME (as well as for the other mentioned protocols). S/MIME uses the Cryptographic Message Syntax (CMS), and Bouncy Castle provides a CMS implementation, which must be adapted to HBS. Fortunately, an Internet-Draft on integration HBS in the CMS already exists [24], and can be used for initial guidance. To further encourage the use of post-quantum S/MIME based on HBS, we intend to wrap the Bouncy Castle HBS S/MIME implementation in a new email client plug-in.

The SSH protocol is used for securing communication and remote login between two hosts. Since OpenSSH is based in part on OpenSSL, the core implementation (independently of the protocol layer) is expected to be straightforward. For SSH integration using OpenSSH, we face cipher suite integration again. The signature part of the cipher suite, which often uses the ECDSA pre-quantum algorithms, will be replaced with a HBS implementation. In addition, once more, a fully post-quantum cipher suite would be sensible. The Diffie-Hellman method or RSA are normally used for SSH key exchange; both are non-quantum-safe. A post-quantum replacement key exchange is therefore needed [20]. Besides, the consequences of situations such as signing key depletion or compromise must be planned for, and public key infrastructures used by the modified protocol adapted in consequence.

4.3 Parameter selection, use cases and performance

Advanced HBS schemes feature numerous parameters. This is beneficial in terms of flexibility, but makes parameter setting decisions more complex. There is no one parameter set that ought to be recommended in general; rather, different parameter sets are optimal for different use cases. We will define a number of different typical use cases and suggest corresponding parameter sets. Research already exists on semi-automated parameter selection. In a 2013 paper [29], Hülsing describes a method to determine optimal parameters for XMSS^{MT} using exact linear optimisation and IBM'S ILOG CPLEX tool [30], a mathematical programming solver. Results are provided for two use cases. A similar approach must be applied to the XMSS variant. Variations between use cases include requirements in terms of signature generation frequency. Signing speed is much more of a factor in scenarios such as TLS connection management than for e.g. software update authentication, were signature generations and verifications are less frequent. A larger, more fine-grained array of use cases should be investigated.

On a related note, in terms of performance improvements, other opportunities such as the use of instruction set extensions have not yet been fully taken advantage of. Consider for instance current work by de Oliveira on faster XMSS implementation using AVX vector instructions [3].

To obtain a concrete assessment of real-world performance for a specific use case, we are implementing an HBS version of an existing update authentication tool. Since the tool is used to verify the integrity of software patches on client systems, the use case is a low-frequency signing scenario. The authentication tool uses the OpenSSL core cryptographic library; RSA signatures are replaced by XMSS / XMSS^{MT} signatures in the post-quantum version.

5 Conclusions

HBS ought to become widely available in security infrastructures to help counter the threat posed by quantum computers. In this paper, we have argued that while HBS are well-understood by the cryptographic research community, many efforts are still required to foster their use in the real world. In practice, we have insisted on the importance of standardisation and integration in common cryptographic software libraries and security protocol implementations. We have also described current work to support these goals. Standards and reference implementations play an ergonomic role for system integrators and facilitate both technical interfacing and strategic decisions such as parameter selection. They must not be neglected if broad adoption is desired. The intensified debate and technical scrutiny resulting from the engagement with standardisation bodies and library developers serves the goal of greater security shared by all stakeholders.

References

1. OpenSSL: The Open Source toolkit for SSL/TLS. `https://www.openssl.org/`
2. Legion of the Bouncy Castle. `https://www.bouncycastle.org/` (2013)
3. Ana Karina D.S. de Oliveira: An efficient software implementation of XMSS. Presented at LatinCrypt 2014 (2014)
4. Bellare, M., Miner, S.K.: A Forward-Secure Digital Signature Scheme. In: CRYPTO '99. Lecture Notes in Computer Science, vol. 1666, pp. 431–448. Springer (1999)

5. Bernstein, D.J., Hopwood, D., Hülsing, A., Lange, T., Niederhagen, R., Papachristodoulou, L., Schneider, M., Schwabe, P., Wilcox-O'Hearn, Z.: SPHINCS: Practical Stateless Hash-Based Signatures. In: EUROCRYPT 2015. Lecture Notes in Computer Science, vol. 9056, pp. 368–397. Springer (2015)
6. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: The KECCAK reference. `http://keccak.noekeon.org/` (2011)
7. Bos, J.W., Costello, C., Naehrig, M., Stebila, D.: Post-quantum key exchange for the TLS protocol from the ring learning with errors problem (2014)
8. Boura, C., Canteaut, A.: Zero-Sum Distinguishers for Iterated Permutations and Application to Keccak-f and Hamsi-256. In: Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 6544, pp. 1–17. Springer (2010)
9. Braun, J., Hülsing, A., Wiesmaier, A., Gagliotti Vigil, M.A., Buchmann, J.: How to Avoid the Breakdown of Public Key Infrastructures — Forward Secure Signatures for Certificate Authorities. In: EuroPKI. Lecture Notes in Computer Science, vol. 7868, pp. 53–68. Springer (2012)
10. Buchmann, J., Dahmen, E., Hülsing, A.: XMSS — A Practical Forward Secure Signature Scheme Based on Minimal Security Assumptions. In: PQCrypto. Lecture Notes in Computer Science, vol. 7071, pp. 117–129. Springer (2011)
11. Buchmann, J., Dahmen, E., Klintsevich, E., Okeya, K., Vuillaume, C.: Merkle signatures with virtually unlimited signature capacity. In: ACNS. Lecture Notes in Computer Science, vol. 4521, pp. 31–45. Springer (2007)
12. Buchmann, J., García, L.C.C., Dahmen, E., Döring, M., Klintsevich, E.: CMSS — An Improved Merkle Signature Scheme. In: INDOCRYPT 2006. Lecture Notes in Computer Science, vol. 4329, pp. 349–363. Springer (2006)
13. Courtois, N., Finiasz, M., Sendrier, N.: How to Achieve a McEliece-Based Digital Signature Scheme. In: ASIACRYPT 2001, Lecture Notes in Computer Science, vol. 2248, pp. 157–174. Springer (2001)
14. DeAngelis, S.F.: Closing In On Quantum Computing. Wired (2014)
15. Ding, J., Schmidt, D.: Rainbow, a New Multivariable Polynomial Signature Scheme. In: Applied Cryptography and Network Security, Lecture Notes in Computer Science, vol. 3531, pp. 164–175. Springer (2005)
16. Dods, C., Smart, N., Stam, M.: Hash Based Digital Signature Schemes. In: Cryptography and Coding, Lecture Notes in Computer Science, vol. 3796, pp. 96–115. Springer (2005)
17. Duc, A., Guo, J., Peyrin, T., Wei, L.: Unaligned Rebound Attack: Application to Keccak. In: Fast Software Encryption. Lecture Notes in Computer Science, vol. 7549, pp. 402–421. Springer (2012)
18. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice Signatures and Bimodal Gaussians. In: CRYPTO 2013, Lecture Notes in Computer Science, vol. 8042, pp. 40–56. Springer (2013)
19. ElGamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In: CRYPTO'84, Lecture Notes in Computer Science, vol. 196, pp. 10–18. Springer (1985)
20. ETSI: White paper: Quantum Safe Cryptography and Security; An introduction, benefits, enablers and challenges. `http://docbox.etsi.org/Workshop/2014/201410_CRYPTO/Quantum_Safe_Whitepaper_1_0_0.pdf` (2014)
21. Gazdag, S., Butin, D.: Practical Hash-based Signatures (Quantencomputer-resistente Signaturverfahren für die Praxis). `http://square-up.org/` (2014)
22. Google: BoringSSL. `https://boringssl.googlesource.com/boringssl/` (2014)
23. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Symposium on Theory of Computing (STOC). pp. 212–219. ACM (1996)

24. Housley, R.: Use of the Hash-based Merkle Tree Signature (MTS) Algorithm in the Cryptographic Message Syntax (CMS). IETF (2014), Internet-Draft
25. Hülsing, A.: Practical Forward Secure Signatures using Minimal Security Assumptions. Ph.D. thesis, Technische Universität Darmstadt (2013)
26. Hülsing, A.: W-OTS+ — Shorter Signatures for Hash-Based Signature Schemes. In: AFRICACRYPT. Lecture Notes in Computer Science, vol. 7918, pp. 173–188. Springer (2013)
27. Hülsing, A., Busold, C., Buchmann, J.: Forward Secure Signatures on Smart Cards. In: Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 7707, pp. 66–80. Springer (2012)
28. Hülsing, A., Butin, D., Gazdag, S.L., Mohaisen, A.: XMSS: Extended Hash-Based Signatures. IETF (2015), Internet-Draft
29. Hülsing, A., Rausch, L., Buchmann, J.: Optimal Parameters for XMSS^{MT}. In: CD-ARES Workshops. Lecture Notes in Computer Science, vol. 8128, pp. 194–208. Springer (2013)
30. IBM: IBM ILOG CPLEX Optimizer. `http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/index.html`
31. Johnson, D., Menezes, A., Vanstone, S.: The Elliptic Curve Digital Signature Algorithm (ECDSA). International Journal of Information Security 1(1), 36–63 (2001)
32. Lamport, L.: Constructing Digital Signatures from a One Way Function. Tech. rep., SRI International Computer Science Laboratory (1979)
33. Lenstra, A.K., Verheul, E.R.: Selecting cryptographic key sizes. J. Cryptology 14(4), 255–293 (2001)
34. McGrew, D., Curcio, M.: Hash-Based Signatures. IETF (2014), Internet-Draft
35. Merkle, R.C.: A Certified Digital Signature. In: CRYPTO. Lecture Notes in Computer Science, vol. 435, pp. 218–238. Springer (1989)
36. National Institute of Standards and Technology: FIPS PUB 186-4: Digital Signature Standard (DSS). National Institute for Standards and Technology (2013), `http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf`
37. Nguyen, P.Q., Regev, O.: Learning a Parallelepiped: Cryptanalysis of GGH and NTRU Signatures. J. Cryptology 22(2), 139–160 (2009)
38. OpenBSD: LibreSSL. `http://www.libressl.org/` (2014)
39. OpenSSL Security Advisory: SSL/TLS MITM vulnerability (CVE-2014-0224). `https://www.openssl.org/news/secadv_20140605.txt` (2014)
40. OpenSSL Security Advisory: TLS heartbeat read overrun (CVE-2014-0160). `https://www.openssl.org/news/secadv_20140407.txt` (2014)
41. Pop, I.M., Geerlings, K., Catelani, G., Schoelkopf, R.J., Glazman, L.I., Devoret, M.H.: Coherent suppression of electromagnetic dissipation due to superconducting quasiparticles. Nature 508(7496), 369–372 (2014)
42. Rich, S., Gellman, B.: NSA seeks to build quantum computer that could crack most types of encryption. The Washington Post (2014)
43. Rivest, R., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM 21, 120–126 (1978)
44. Saeedi, K. et al: Room-Temperature Quantum Bit Storage Exceeding 39 Minutes Using Ionized Donors in Silicon-28. Science 342(6160), 830–833 (2013)
45. Shor, P.W.: Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM J. Comput. 26(5), 1484–1509 (1997)
46. TU Darmstadt: FlexiProvider, an open source Java Cryptographic Service Provider. `http://www.flexiprovider.de/javadoc/flexiprovider/docs/de/flexiprovider/pqc/hbc/gmss/package-summary.html` (2006)